A. Gradient alignment and sparsity in SiM4C

A.1. Gradient alignment in SiM4C

Nichol et al. [38] analyzed the meta-gradients from the MAML objective [13] after k inner optimization steps. Following the notation introduced in Section 2.2, let $L^{inn}(\theta^{i-1})$ and $L^{out}(\theta^k)$ be the loss at the i^{th} inner step, and the outer loss, respectively. Then, they showed that via a second-order Taylor expansion, the meta-gradients can be approximated by the expression:

$$\frac{\partial L^{MAML}(\theta)}{\partial \theta} \approx \frac{\partial L^{out}(\theta^k)}{\partial \theta^k} - \eta \sum_{i=0}^{k} \frac{\partial}{\partial \theta^i}\left(\frac{\partial L^{inn}(\theta^i)}{\partial \theta^i} \cdot \frac{\partial L^{out}(\theta^k)}{\partial \theta^k}\right) + \frac{\partial}{\partial \theta^k}\left(\frac{\partial L^{inn}(\theta^i)}{\partial \theta^i} \cdot \frac{\partial L^{out}(\theta^k)}{\partial \theta^k}\right). \tag{8}$$

The last two terms in Equation 8 (*green*) correspond to the *gradient of the dot product* between the *gradient of each inner step* and the *outer gradient*. Hence, optimizing the MAML loss approximately corresponds to *maximizing gradient alignment* between the gradient of each inner step and the gradient of the outer loss. Intuitively, if two losses have gradients with a positive dot product, taking a small enough step in the direction of either of their gradients will locally minimize them both. Thus, gradient alignment between different objectives induces *transfer*, incentivizing feature reuse when computing losses with different inputs if and only if the underlying gradients point in similar directions [35, 41]. In the context of SiM4C, during meta pre-training, $\frac{\partial L^{inn}(\theta)}{\partial \theta}$ is the gradient of a random task's loss using a single random data point while $\frac{\partial L^{out}(\theta')}{\partial \theta'}$ is the gradient of the continual objective from an i.i.d. batch of task samples (past data) and unseen current task samples (future data). Hence, optimizing gradient alignment in SiM4C induces f_θ to structure its latent representations such that optimizing from an arbitrary sample is likely to not interfere with past learning and generalize to unseen task samples, given the observed distribution of tasks.

Nichol et al. [38] showed that the second-order Taylor approximation also holds with first-order approximations to the meta-gradient (e.g., FOMAML) with some caveats. In particular, it only holds *in expectation* over the meta-loss, and the magnitude of the gradient alignment term is strictly lower than the gradient alignment term in Equation 8. We note that both implementations of OML and ANML rely on a FOMAML approximation, given their large k, and use a meta-batch size of one[3]. Hence, the aforementioned caveats might theoretically hinder their effectiveness for maximizing gradient alignment. Moreover, while SiM4C's outer gradient is calculated mostly from *unseen* samples, ANML's outer loss only considers *seen* samples from the current task and the remember set, hence, potentially leading to poor gradient alignment with respect to unseen data. This intuition appears to be consistent with our empirical analysis from Section 4, potentially providing a complementary explanation for ANML's lower test generalization and the occurrence of meta-overfitting.

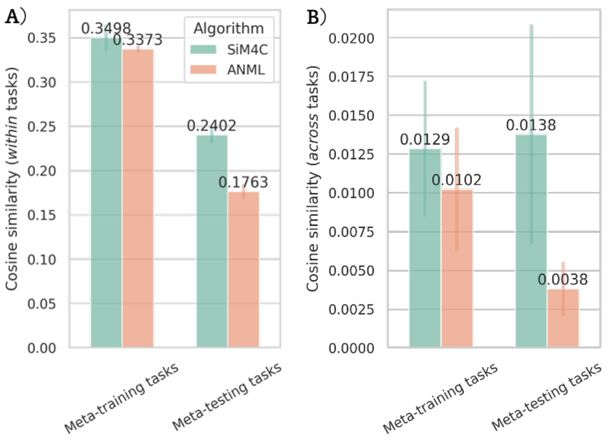

Figure 4. Cosine similarity measured **(A) within** and **(B) across** different tasks collected throughout performing continual learning on meta pre-trained models using 600 sampled tasks either from the meta-training or the unseen meta-testing set. The experiments are conducted on the online continual learning Omniglot benchmark with meta pre-training, as described in Section 3.1.

Recording gradient alignment. We further analyze if the aforementioned considerations and hypotheses empirically hold. In particular, we collect measures of alignment and transfer using the cosine similarity between different gradients at meta-testing. We compare ANML and SiM4C, evaluating both *within* task alignment and *across* tasks alignment. These are computed as the cosine of the angle between different gradients with respect to new samples within the same meta-testing task or between the gradients of each new sample and a batch of test data from all meta-testing tasks. We show our results in Figure 4, where we illustrate the mean alignments when deploying the meta pre-trained classifier on either the meta-testing tasks or the same tasks observed during meta pre-training. On the meta-testing classes, SiM4C achieves comparatively higher cosine similarity as compared to ANML, both within the same task and across tasks. In contrast, our recordings from performing continual learning reusing the meta pre-training tasks, show the two algorithms achieve comparable cosine similarities. These results appear to closely reflect the above considerations motivating SiM4C's implementation, empirically confirming that multiple inner steps are not needed to achieve backward and forward transfer. Moreover, they also

[3] In each iteration a single task is used to calculate the meta-gradient

validate our hypothesis that the emphasis on unseen data in SiM4C's meta-objective allows gradient alignment to occur for tasks beyond the meta-training set, and provide further evidence for the meta-overfitting of prior methods.

A.2. Induced representation sparsity

As argued by Javed and White [23], following the intuition proposed by French [14], representation sparsity appears to be a useful heuristic to attain representations that avoid catastrophic forgetting. As an illustrative example, if different data points x_i, x_j from unrelated tasks do not share any of their final activations z_i, z_j, then the gradients with respect to the final weight matrix W will not interfere with one another. This occurs since the gradient corresponds to a matrix product between a column and row vector:

$$\frac{\partial L(f_\theta(x))}{\partial W} = z^T \frac{\partial L(f_\theta(x))}{\partial f_\theta(x)}. \tag{9}$$

Thus, $\frac{\partial L(f_\theta(x))}{\partial W}$ could have non-zero rows only where z has non-zero entries. It hs been empirically shown that just inducing sparsity as a heuristic auxiliary objective appears to incur in *dead neurons*, where certain parts of the representations are never activated, resulting in a loss of capacity. Consistently with the empirical findings of prior meta pre-trained continual learning work [4, 23], we hypothesize that also the representations of SiM4C after meta pre-training will be sparse as an emergent side effect of improved knowledge-retention.

Method	**Active neurons**	**Dead neurons**
SR-NN [34]	15%	0.7%
OML [23]	3.8%	0%
ANML [4]	5.9%	0%
SiM4C	10.7%	0%

Table 2. Percentages of active neurons (magnitude above 1% the mean activation) and dead neurons (never active) of the final representations computed with the meta-testing samples for different meta pre-training techniques.

Hence, we compare the sparsity levels achieved by OML, ANML, SiM4C, and an alternative method considered by Javed and White [23] using the set-KL method as a way of achieving sparse representations (SR-NN) [34]. Following Beaulieu et al. [4], we consider an activation *active* if its magnitude is greater than 1% of the average activation in the representation. We directly report all baseline sparsity results obtained in [4, 23]. As shown in Table 2, SiM4C's representations are inherently sparse with, on average, only 10.7% neurons active for any given data point. Moreover, while its representations are considerably more sparse than the SR-NN baseline, SiM4C does not experience any dead neurons, validating that it does not waste representational capacity as opposed to methods that optimize directly for sparsity heuristics. Our results further reinforce Javed and White [23]'s hypothesis that sparsity is an *emergent* property of effective representations for continual learning. However, we note that SiM4C's representations are comparatively *less* sparse than OML and ANML. Yet, this does not come at either performance or gradient alignment costs, as shown in Section 4 and Appendix A.1 above. We believe this to be a consequence of SiM4C's objective incentivizing not only to minimize forgetting but also to maximize *positive* transfer to large amounts of unseen 'future' data. In fact, while sparse representations avoid interference, they also prevent potentially positive transfer between learning different samples, as the gradients from all inactive features will tend to zero. Therefore, we argue that excessive sparsity is likely detrimental and a symptom of the meta-learned model being over-conservative about stored information, leading to lower transfer and worse generalization, as shown by our results.

In Figure 5, we also provide some visualizations of SiM4C's 2304-dimensional normalized final activations from random meta-testing samples, reshaped into 32×72 heatmaps. We compare them with the average normalized activation across all samples. The sampled visualizations exemplify SiM4C's sparsity, while the uniformity in the average normalized activations reflects SiM4C's lack of dead neurons and its efficient use of representational capacity across different tasks.

B. Implementation details

B.1. SiM4C with meta pre-training

Our implementation of SiM4C for the meta pre-trained continual learning problem setting follows the model, preprocessing, and training paradigm from Javed and White [23] that was later refined by Beaulieu et al. [4]. We build off our method from a close re-implementation of their own code to make sure we evaluate the merits of the different optimization strategies as opposed to auxiliary details (e.g., network architecture). We provide an overview in Table 3 and a thorough description below.

Architecture details. The original code implementation of Beaulieu et al. [4] starts by pre-processing the inputs by reshaping the 84×84 Omniglot images to 28×28 and normalizing them to unit mean (although this step does not appear to be important since inputs are already binary). The utilized network architecture for the continual classifier f_θ is a simple convolutional network. The network is split into a representation network (parameterized by the representation weights θ_r) and a final prediction head (parameterized by the prediction weights θ_p). The representation network

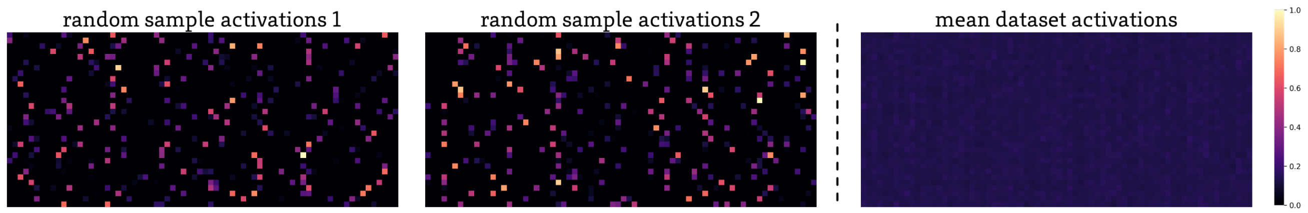

Figure 5. Heatmaps of SiM4C's final activations reshaped into 32×72 matrices, computed from two random data points sampled from all the meta-testing tasks. In the rightmost heatmap, we also show the mean activations across all the meta-testing data points.

Meta pre-trained SiM4C hyper-parameters	
Convolutional layers	3
Convolution filters	256
Convolution filter size	3×3
Nonlinearity	ReLU
Normalization	Instance normalization [47]
Maxpooling layers	2
Maxpooling filter size	2×2
Inner optimizer	Gradient descent
Inner optimizer learning rate	0.1
Meta optimizer	Adam [24]
Meta learning rate	0.001
Meta-testing optimizer	Adam [24]
Meta-testing learning rate	0.001
Meta-gradient	MAML [13]
Total number of updates	100000
Save model every	5000
Total sampled samples per task	20
Inner steps	1
Remember set size	64

Table 3. SiM4C hyper-parameters on the Omniglot online continual learning with meta pre-training benchmark. Choices try to be consistent with Beaulieu et al. [4] and Javed and White [23].

comprises three convolutional layers (with 256 filters of size 3×3), each followed by an instance normalization layer [47], and a ReLU activation function. The first two layers are also followed by a maxpooling layer with a filter dimension of 2×2. Then the representations are flattened before being processed by the prediction network, comprising a single linear layer. Beaulieu et al. [4] also employs a parallel *Neuromodulatory* network with a similar architecture and even more total parameters, which meta-learns to suppress the final representations of the representation network. As this is their core distinguishing feature from OML and adds non-trivial complexity, we do not employ this network with SiM4C.

Optimization details. Again, following Beaulieu et al. [4], we perform the outer (meta) optimization of θ with an Adam [24] optimizer with default hyper-parameters (i.e., 0.001 learning rate, 0.9 momentum coefficient). Instead, inner optimization on the prediction weights θ_p is carried out with standard gradient descent with a learning rate of 0.1 and no momentum. Interestingly, at meta-testing time, the implementation of Beaulieu et al. [4] uses Adam rather than SGD to finetune the prediction weights. While this appears to be counter-intuitive, we kept this detail also in our implementation for consistency. Unlike ANML and OML, we employ the exact second-order gradient from MAML [13] rather than first-order approximations. We meta pre-train the model for 100K steps (as opposed to OML's 200K) as we observed no notable performance gains after this threshold. We saved checkpoints from each model after 5000 steps. While SiM4C's performance was very robust to the checkpoint weights utilized for meta-testing, the other algorithms appeared to be very sensitive to this choice, with meta-testing performances decreasing by more than 15% absolute accuracy if meta pre-training for too long. Since we could not find any detail for ANML and OML about how early stopping was implemented from their papers, we just reported their best-performing checkpoint at meta-testing (which is what appears to be done also for other hyper-parameter selections from their shared implementations). Hence, we note this is likely overestimating their actual performance for arbitrary continual learning applications. It also exemplifies the practical usefulness of a more robust optimization procedure as SiM4C. As in ANML's meta-training phase, we also sample 20 (the maximum) different samples from a sampled task and employ a remember set of size 64. However, we perform a single inner optimization step, as proposed in our alternative algorithm. Finally, during meta pre-training, we reset the final classification weights of the sampled task to avoid prior meta-learning iterations providing information to the prediction network, as consistently also carried out by ANML and OML. In preliminary experiments, we also tested resetting to their initial value the classification weights of additional randomly sampled classes. While we found this to potentially further help against possible gradient vanishing problems, we did not experience any significant boost in peak performance for any of the considered algorithms.

Buffer	**Method**	**S-CIFAR-10**	**S-Tiny-ImageNet**
200	ER with SiM4C	$lr = 0.03$	$lr = 0.03$
	DER with SiM4C	$lr = 0.03, \alpha = 0.1$	$lr = 0.03, \alpha = 0.1$
	DER++ with SiM4C	$lr = 0.03, \alpha = 0.1, \beta = 0.5$	$lr = 0.03, \alpha = 0.1, \beta = 0.5$
500	ER with SiM4C	$lr = 0.03$	$lr = 0.03$
	DER with SiM4C	$lr = 0.03, \alpha = 0.1$	$lr = 0.03, \alpha = 0.1$
	DER++ with SiM4C	$lr = 0.03, \alpha = 0.1, \beta = 0.5$	$lr = 0.03, \alpha = 0.1, \beta = 0.5$
5120	ER with SiM4C	$lr = 0.03$	$lr = 0.03$
	DER with SiM4C	$lr = 0.03, \alpha = 0.1$	$lr = 0.03, \alpha = 0.1$
	DER++ with SiM4C	$lr = 0.03, \alpha = 0.1, \beta = 0.5$	$lr = 0.03, \alpha = 0.1, \beta = 0.5$
Buffer	**Method**	**P-MNIST**	**R-MNIST**
200	ER with SiM4C	$lr = 0.1$	$lr = 0.1$
	DER with SiM4C	$lr = 0.1, \alpha = 1$	$lr = 0.1, \alpha = 1$
	DER++ with SiM4C	$lr = 0.1, \alpha = 1, \beta = 1$	$lr = 0.1, \alpha = 1, \beta = 1$
500	ER with SiM4C	$lr = 0.1$	$lr = 0.1$
	DER with SiM4C	$lr = 0.1, \alpha = 1$	$lr = 0.1, \alpha = 1$
	DER++ with SiM4C	$lr = 0.1, \alpha = 1, \beta = 1$	$lr = 0.1, \alpha = 1, \beta = 1$
5120	ER with SiM4C	$lr = 0.1$	$lr = 0.1$
	DER with SiM4C	$lr = 0.1, \alpha = 1$	$lr = 0.1, \alpha = 1$
	DER++ with SiM4C	$lr = 0.1, \alpha = 1, \beta = 1$	$lr = 0.1, \alpha = 1, \beta = 1$

Table 4. SiM4C hyper-parameters on the Class-IL, Task-IL, Domain-IL benchmarks. Choices are consistent across S-CIFAR-10 and S-TINY-Imagenet, and across P-MNIST and R-MNIST. In particular, for both these groups of experiments we use the same learning rate, α, and β values.

B.2. SiM4C without meta pre-training

In our implementation of SiM4C for continual learning without meta pre-training, we integrate it with the main memory-based algorithms considered by Buzzega et al. [6]. To show the generality of our procedure, as described in Section 5, we add the proposed SiM4C meta-learning loss after a single inner step as an auxiliary loss *without any scaling coefficient*. Thus, we remark that **SiM4C does not introduce any additional hyper-parameter**. While adding complexity and tuning can provide performance benefits, they can be easily confounded with *overfitting* to a particular set of problems. Moreover, depending on the metric chosen to select hyper-parameters, they can even implicitly break important continual learning constraints about task accessibility [11].

Shared training and architecture details. We build our implementation on top of the code from [6] and use exactly the same models, training procedures, and data augmentation strategies for all experiments, ensuring that we specifically isolate the contributions from SiM4C. As mentioned in Section 5, for the Class-IL and Task-IL experiments on S-CIFAR-10 and S-Tiny-ImageNet, all methods employ a standard ResNet18 architecture [19]. In each newly encountered task, we train for 50 epochs in the S-CIFAR-10 experiments and 100 epochs in the S-Tiny-ImageNet experiments. In both benchmarks, 32 data points are sampled from the memory buffer and the current task at each training step, with a total batch size of 64. The used augmentation strategy involves a combination of random cropping and flipping the inputs. Instead, for the Domain-IL experiments on the lower-dimensional Permuted MNIST and Rotated MNIST datasets, all methods employ a simple fully connected network with two layers and 100 hidden features. ReLU activations without normalization are employed between the layers. Training for each new task is carried out in a single epoch sampling 10 data points from both the task and the memory buffer data (i.e., with a total batch size of 20). No image augmentation is used. In all aforementioned benchmarks, the models are trained with a standard SGD optimizer with a tuned learning rate.

Obtaining the *future data batch*. As introduced in Section 5, SiM4C requires an additional batch of *future data* from the current task to carry out its auxiliary optimization. To avoid incurring additional memory costs from using multiple batches of sampled data in each optimization step, we opted for the simple solution of simply re-augmenting the provided batch of data an additional time. In practice, we found this simple strategy to work almost as well. Moreover, since in the Domain-IL experiments, no data augmentation is applied, we opted for the even simpler solution of *splitting* the 10 samples from the current task in two different smaller batches of 5 samples each, used as the current data (for the inner step) and the future data (for the outer step), respectively. The reason we did not apply this simple idea also for the Class-IL and Task-IL experiments on S-CIFAR-10 and S-Tiny-ImageNet, is that the utilized ResNet18 entails several batch normalization layers [22], for which we wanted to maintain the number of past, present, and future samples consistent. However, as shown in the later ablation we performed (Appendix E), the actual downsides of using different batch sizes are not considerable, and all three mentioned choices of data selection perform comparably on almost all problem settings, as long as they provide some sensible information of within-task variation.

Hyper-parameters. Buzzega et al. [6] performs extensive hyper-parameter tuning with grid search on a held-out subset of data to which it assumes to have unrestricted access (arguably breaching the continual learning canonical restriction). They tune all the key parameters of every method they consider, including the learning rate, and repeat this for each different benchmark and memory buffer size considered. We refer to Tables 9 and 10 in their paper for all details. Instead, we find that our integration with SiM4C does not require any major tuning for performance, selected based on the default values for ER, DER, and DER++. In fact, unlike for the reported baseline scores, we obtain our results using consistent hyper-parameters across multiple datasets, base algorithms, and memory buffer sizes. The only required modification was to increase the learning rate and α for the Domain-IL experiments to balance training being now over a single epoch per task. We provide the full list of values (with redundant entries) in Table 4. We believe that using tuned baselines and avoiding tuning our own implementations is strong evidence for the added robustness provided by integrating SiM4C, given the substantial gains recorded in Section 5.

B.3. Hardware setup

We run our experiments on a combination of different local and remote hardware. The remote hardware comprises NVIDIA's *MAXQ Deep Learning Systems* with NVIDIA V100 GPUs. We also employ a local server with two NVIDIA RTX 3090 GPUs and an AMD Ryzen Threadripper 3970x CPU. However, to ensure our recorded memory utilization and optimization timings are consistent, we collect also recordings from each experiment individually on another local server with a single NVIDIA Titan XP GPU and an Intel Core i7-7700K CPU.

C. Extended results

C.1. Additional integrations and comparisons

In this Section, we extend the results from Section 5, by reporting the Class-IL, Task-IL, and Domain-IL performance of the tested algorithms and baselines with an additional memory buffer size of 5120. Moreover, we also compare also with the other popular non-memory-based continual learning algorithms [33, 43, 46, 55]. We note that while effective, some of these baselines (e.g., Progressive Neural Networks (PNN) [43]) make strong assumptions about access to specific task boundaries, and cannot be directly applied to the more general Class-IL and Domain-IL settings. We refer to Section 5 and Appendix B for all details about our experiments on these benchmarks.

As shown in Table 5, SiM4C once again provides memory-based algorithms with near-universal improvements in performance, achieving state-of-the-art results also for the considered increased buffer size. Moreover, the largest gains are attained in the more challenging problem settings precluding task information (Class-IL) or involving increased amounts of more complex samples (S-Tiny-ImageNet). Even in the comparatively easier Domain-IL settings on the MNIST variants, we again recorded performance benefits (with a lower magnitude) in all but one experiment. Expectedly, relative gains appear to diminish with increased buffer sizes, as the different base algorithms already have a smaller gap with the i.i.d. performance of joint training, leaving a smaller room for improvements.

C.2. Per-task detailed performance curves

We further extend the results from Section 5 with more fine-grained metrics showing the *per-task* performance curves of the DER algorithm, with and without SiM4C on the S-CIFAR-10 dataset. In Figure 6 we show how performance evolves for each of the five tasks comprising two of the CIFAR10 classes with different memory buffer sizes and in both Class-IL and Task-IL settings. Integrating SiM4C appears to ease the integration of new information, as classification accuracy for the new tasks is consistently higher, indicating better forward transfer. Moreover, the overall amount of forgetting, especially for the earlier tasks is reduced, indicating better information retention and backward transfer on longer optimization horizons.

C.3. Forgetting and forward transfer

In Table C.3 we further extend the results from Section 5 by also providing the amounts of forgetting and forward transfer as measured by Buzzega et al. [6]. We only report forward transfer in the Domain-IL continual learning problems, since there is no principled and precise metric to estimate forward transfer when the classes in each task are mutually exclusive. However, since performance on the

Buffer	Method	S-CIFAR-10		S-Tiny-ImageNet		P-MNIST	R-MNIST
		Class-IL	*Task-IL*	*Class-IL*	*Task-IL*	*Domain-IL*	*Domain-IL*
✗	JOINT	92.20	98.31	59.99	82.04	94.33	95.76
	SGD	19.62	61.02	7.92	18.31	40.70	67.66
✗	oEWC [46]	19.49	68.29	7.58	19.20	75.79	77.35
	SI [55]	19.48	68.05	6.58	36.32	65.86	71.91
	LwF [33]	19.61	63.29	8.46	15.85	-	-
	PNN [43]	-	95.13	-	67.84	-	-
5120	ER [41]	82.47	96.98	27.40	67.29	89.90	93.45
(Ours)	**ER with SiM4C**	85.52 +3.7%	**97.13** +0.2%	28.78 +5.0%	68.63 +2.0%	89.66 −0.3%	93.98 +0.6%
	GEM [35]	25.26	95.55	-	-	87.42	88.57
	A-GEM [8]	21.99	90.10	7.96	26.22	73.32	80.18
	iCaRL [40]	55.07	92.23	14.08	40.83	-	-
	FDR [5]	19.70	94.32	28.97	68.01	90.87	94.19
	GSS [2]	67.27	94.19	-	-	82.22	85.24
	HAL [9]	59.12	88.51	-	-	89.20	91.17
	DER [6]	83.81	95.43	36.73	69.50	91.66	94.14
(Ours)	**DER with SiM4C**	86.03 +2.6%	96.00 +0.6%	39.61 +7.8%	69.55 +0.1%	91.74 +0.1%	94.34 +0.2%
	DER++ [6]	85.24	96.12	39.02	69.84	92.26	94.65
(Ours)	**DER++ with SiM4C**	**86.05** +1.0%	96.45 +0.3%	**40.27** +3.2%	**70.06** +0.3%	**92.35** +0.1%	**94.70** +0.1%

Table 5. Mean final classification accuracy and relative improvements (*green*) from using SiM4C with different memory-based algorithms, as evaluated on popular continual learning benchmarks on an additional memory buffer size. We provide the baseline results reported in prior work including also other non-memory-based baselines. Empty entries indicate either incompatibility with the relative problem setting (LwF, PNN, iCaRL in Domain-IL) or intractable training times [6].

		A) FORGETTING				B) FWD. TRANSFER	
Buffer	Method	S-CIFAR-10		P-MNIST	R-MNIST	P-MNIST	R-MNIST
		Class-IL	*Task-IL*	*Domain-IL*	*Domain-IL*	*Domain-IL*	*Domain-IL*
200	ER	61.24	7.08	22.54	8.87	**1.37**	66.79
	DER	40.76	6.57	14.00	6.53	1.23	64.69
	DER++	32.59	5.16	11.49	6.08	0.91	**67.05**
	DER with SiM4C	**26.57**	**4.10**	**9.95**	**5.62**	1.19	67.00
500	ER	45.35	3.54	14.90	8.02	**0.56**	65.52
	DER	26.74	4.56	8.07	3.96	0.21	**72.45**
	DER++	22.38	4.66	7.67	**3.57**	−0.35	67.05
	DER with SiM4C	**21.54**	**3.38**	**7.58**	3.71	0.33	72.21

Table 6. Detailed results with **(A) forgetting**, lower is better, and **(B) forward transfer**, higher is better. We compare the main algorithms from Buzzega et al. [6] with our integration of DER with SiM4C.

MNIST-based continual learning tasks already appears to be quite saturated, these results are likely less informative. In line with our intuition and previous observations, overall, adding SiM4C on top of DER lowers forgetting and improves forward transfer.

C.4. Training time and memory

We also provide recordings of the time taken for a single training iteration, together with GPU memory requirements of our memory-based SiM4C integrations and the relative base algorithms employed. As shown in Figure 7, using SiM4C predictably adds some time and memory overheads to the base algorithms, since it introduces a new separate step in the optimization procedure. However, we see these overheads appear to be quite limited and comparable with the overheads of other tractable non-meta-learning advances. For instance, the recorded time overhead of DER++ from DER appears to be greater than introducing SiM4C to DER. Also, GPU memory costs appear to be dominated by other processes in the optimization procedure even after the introduction of SiM4C. The tractability of SiM4C is due to its minimal effective design, avoiding the introduction of

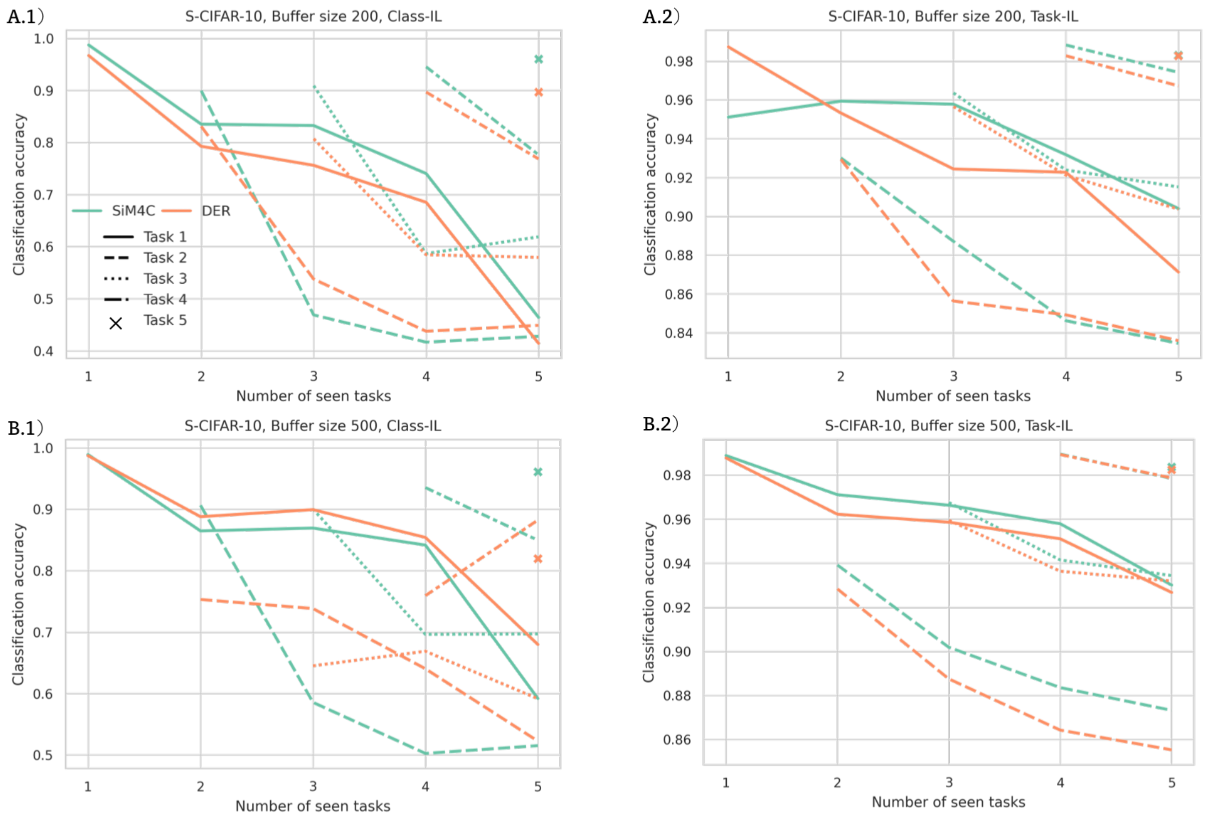

Figure 6. Detailed results comparing the per-task performance curves of DER and our integration with SiM4C on the S-CIFAR-10 dataset. We show how the classification accuracy evolves for the classes in each of the five tasks, when using **(A) memory buffers of size 200** and **(B) memory buffers of size 500**, in both **(1) Class-IL** and **(2) Task-IL** continual learning settings.

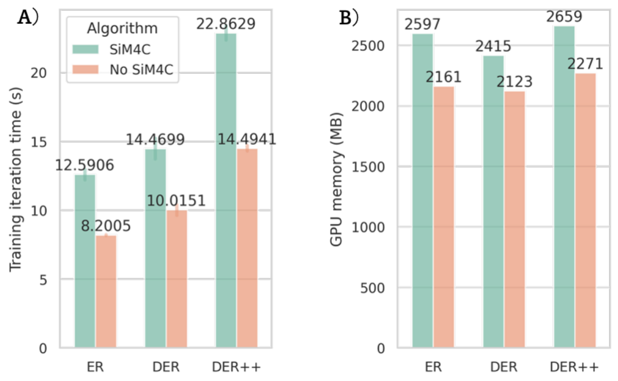

Figure 7. **(A) average time taken** to perform a single training iteration and **(B) GPU memory requirements**, as measured on the S-CIFAR-10 experiments with our different integrations of SiM4C. We compare with the results for the relative base algorithms used in our integrations.

large computation graphs that often occur when traditionally applying meta-learning for continual learning [17, 41].

D. SiM4C ablations and design

In this Section, we evaluate several modifications to SiM4C as a meta pre-training procedure, to validate and understand the effectiveness of its design choices. We again make use of the Omniglot dataset and follow the same meta pre-training and meta-testing procedures details in Section 3.1 and Appendix B. To summarize our results, we report them in a tabular format, showing the final classification accuracy at the end of meta-testing on 600 tasks both with respect to the test and training data. We also report the different algorithms' GPU memory utilization and time to perform a single optimization loop to understand their tradeoffs. As in the rest of our experiments, we use five different random seeds.

D.1. SiM4C objective ablations

We validate the efficacy of three distinct aspects of SiM4C's optimization procedure: the exact second-order gradient computation, together with both the utilization of future and past data in the meta-loss. In particular, we evaluate using the same first-order gradient approximation with SiM4C as in prior work [4, 17, 23, 41], not including in

Method	**Final test accuracy**	**Final training accuracy**	**GPU memory (MB)**	**Optimization time (ms)**
SiM4C	$0.713_{\pm 0.00}$	$0.972_{\pm 0.00}$	1487	$0.118_{\pm 0.01}$
		SiM4C meta-objective ablations		
First order gradient	$0.692_{\pm 0.01}$	$0.960_{\pm 0.00}$	1472	$0.116_{\pm 0.02}$
No future data	$0.709_{\pm 0.00}$	$0.974_{\pm 0.00}$	1485	$0.119_{\pm 0.01}$
No past data	$0.004_{\pm 0.00}$	$0.012_{\pm 0.01}$	1481	$0.109_{\pm 0.02}$
		SiM4C with increased inner steps		
2 inner steps	$0.714_{\pm 0.00}$	$0.974_{\pm 0.00}$	1498	$0.157_{\pm 0.01}$
5 inner steps	$0.709_{\pm 0.00}$	$0.972_{\pm 0.00}$	1525	$0.228_{\pm 0.01}$
10 inner steps	$0.702_{\pm 0.00}$	$0.968_{\pm 0.00}$	1542	$0.313_{\pm 0.00}$
20 inner steps	$0.680_{\pm 0.00}$	$0.976_{\pm 0.00}$	1603	$0.401_{\pm 0.00}$

Table 7. Comparison of different modifications of our SiM4C meta pre-trained implementation. We report the mean and standard deviation of the final performance on both test and training samples at the end of meta-testing, after experiencing 600 tasks on the Omniglot online continual learning benchmark. We also report the recorded GPU memory and optimization time of each algorithm.

the meta loss any unseen 'future data' from the current task and not including any 'past data' from all previous classes (i.e., ablating the remember set). We provide the results in the top half of Table 7. All three ablations degrade test performance, validating our methodology. Not using past data appears to have the most drastic effect, as without a remember set the meta-objective does not incentivize the model to remember old tasks, but only to optimize the current task. Hence, the only incentive left in the meta-objective is to discard all prior knowledge and learn how to best quickly solve the current task in a single step, potentially even exacerbating catastrophic forgetting. We would also like to note that the performance difference with the ablating future data from the meta-objective is notably lower than from using first-order gradients. This can be explained by the fact that since we are only taking a single inner gradient step for each task, with a single data point out of 20 available in Omniglot, the remember set will already contain a lot of unseen samples, including from the current task. Thus, including the rest of the unseen task data is likely to have a still positive, but lower-magnitude impact on performance as opposed to scenarios where meta pre-training task data was more limited.

D.2. Increasing the number of inner steps

We also validate the efficacy of another fundamental design principle of SiM4C, the number of inner steps k. In particular, we meta pre-train SiM4C with $k = 2, k = 5, k = 10$, and $k = 20$ inner steps. We provide the results in the bottom half of Table 7. As expected, increasing k leads to a monotonic increase in GPU memory allocation and optimization time, thus, increasing the computational cost of meta pre-training. Furthermore, we see that increasing the value of k beyond 2 leads to an increasing deterioration of test accuracy. We hypothesize that two potentially complementary factors explain this deterioration,

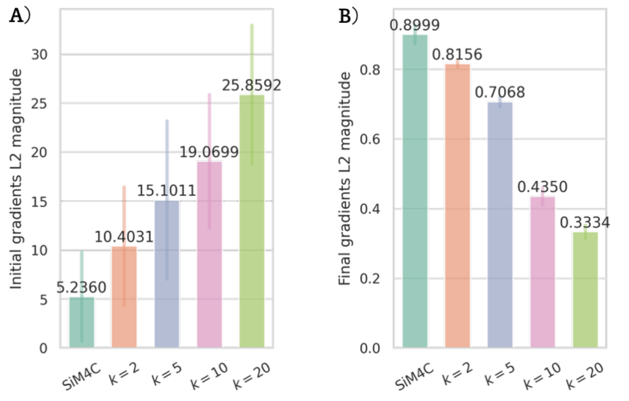

Figure 8. Average L2 gradient norms recorded in the **(A) first** and **(B) last** 500 meta pre-training iterations of SiM4C with different numbers of inner steps k used in the meta-objective.

in line with previous considerations. First, a higher k leads to lower amounts of unseen 'future' data in the meta-loss as the model will have seen greater amounts of the available samples for both the current and past tasks. Hence, the resulting meta-learning optimization will place increasingly less emphasis on future transfer and generalization. This phenomenon appears to be evidenced by the fact that, unlike test accuracy, the training accuracy on the seen training examples does not seem to be majorly affected, showing that meta pre-training with larger values of k does not degrade knowledge retention with respect to seen past samples. Second, backpropagating through a large number of inner steps appears to hinder the stability of meta pre-training, consistently with some of the observations of Antoniou et al. [3]. The reason for these instabilities comes from the repeated forward and backward passes through the same layers of a neural network during the inner optimization process, facilitating the emergence of exploding and vanishing effects.

We provide evidence of this phenomenon occurring during meta pre-training by recording and comparing the L2 magnitude of the meta-gradients during both the first and the last 500 iterations of meta pre-training. As shown in Figure 8, during the beginning of meta pre-training, increasing the number of inner steps leads to an *increase* up to five times an already high gradient magnitude. Instead, at the end of meta pre-training, increasing the number of inner steps instead leads to a *decrease* of an already much lower gradient magnitude. Concretely, the ratio of start to final gradient magnitudes is less than 6 for the original $k = 1$ and over 78 for $k = 20$, evidencing the increased sensitivity to the aforementioned instabilities.

D.3. Data diversity for meta-pre-training

We further extend our comparison of ANML and SiM4C by analyzing how pre-training data diversity affects the effectiveness of these algorithms. In particular, we compare their final classification accuracy after meta-testing on the full set of 600 unseen tasks, after meta-pre-training with a reduced number of total tasks (Figure 9 A) and a reduced number of total samples per task (Figure 9 B). Predictably, for both algorithms, diminishing data diversity monotonically hiders test performance. However, SiM4C is visibly less affected than ANML. For instance, even under the most extreme considered settings (100 meta-pre-training tasks and 5 samples per task) SiM4C still recovers a final classification accuracy of around 60%, very close to ANML's performance using the full set of meta-pre-training data. In contrast, ANML's final classification accuracy sinks, getting as low as 21%, likely due to meta-overfitting to the simplified meta-pre-training setting. These results highlight the difficulty of meta-learning under constrained data budgets and also further highlight SiM4C effectiveness and ability to overcome meta-overfitting.

E. Auxiliary meta-optimization and future data generation strategies

As detailed in Section 5, when integrated with memory-based algorithms, SiM4C optimizes its auxiliary meta-objective and relies on what we refer to as a 'future' batch of data D^{fut} from the current task. In this section, we evaluate different modification to this auxiliary optimization step. We introduced several strategies to obtain both the 'current' batch of samples D^{curr} (to use in the inner optimization step) and this 'future' batch of samples from a single batch of data available at each training step. The employed strategy in most of our main experiments is to make use of the data-augmentation strategies already-present in the considered continual learning baselines, for which the precise scope is to capture within-task variations. Hence, we augment the sampled batch from the current task two independent times, to act as both D^{curr} and D^{fut}. We will refer to this approach as **SiM4C (re-augment)**. However, we believe that a more natural choice would entail temporarily storing each sampled batch data of the current task for two optimization steps rather than one. This would allow for the earlier collected batch to act as D^{curr} and the later collected batch to act as D^{fut}. However, we note that this approach introduces some additional minimal constant memory costs from storing the extra batch. Hence, we do not employ this strategy in our main experiments to ensure a fair comparison with the other memory-based algorithms. We will refer to this approach as **SiM4C (past data)**. Furthermore, as detailed in Appendix B.2, as in the MNIST Domain-IL experiments no data augmentation is applied, we simply split the 10 samples from the current task into two different smaller batches of 5 samples each to act as D^{curr} and D^{fut}. However, for the other benchmarks, using a smaller batch size for the current and future data and a standard batch size for the past data sampled from the memory buffer could incur issues

Buffer	**Method**	**S-CIFAR-10**	
		Class-IL	*Task-IL*
200	ER [41]	44.79	91.19
	ER with SiM4C (no meta)	53.79	92.41
	ER with SiM4C (re-augment)	54.77	92.38
	ER with SiM4C (past data)	54.73	**92.43**
	ER with SiM4C (split data)	56.02	91.96
	ER with SiM4C (grad. split)	**57.24**	92.15
	DER [6]	61.93	91.40
	DER with SiM4C (no meta)	61.47	91.86
	DER with SiM4C (re-augment)	63.81	90.85
	DER with SiM4C (past data)	**63.85**	**92.19**
	DER with SiM4C (split data)	54.34	92.15
	DER with SiM4C (grad. split)	59.09	91.74
500	ER [41]	57.74	93.61
	ER with SiM4C (no meta)	61.02	90.75
	ER with SiM4C (re-augment)	66.05	94.37
	ER with SiM4C (past data)	66.61	**94.42**
	ER with SiM4C (split data)	66.32	93.72
	ER with SiM4C (grad. split)	**67.52**	92.84
	DER [6]	70.51	93.40
	DER with SiM4C (no meta)	65.06	94.01
	DER with SiM4C (re-augment)	**72.62**	93.94
	DER with SiM4C (past data)	72.49	93.89
	DER with SiM4C (split data)	63.73	**94.03**
	DER with SiM4C (grad. split)	67.29	93.99

Table 8. Mean final classification accuracy from ablating meta gradients from the SiM4C auxiliary loss and different strategies to obtain both a 'current' and 'future' data batch (in brackets). We focus on the S-CIFAR-10 benchmark in Class-IL and Task-IL settings for the DER and ER-based algorithms with two memory buffer sizes.

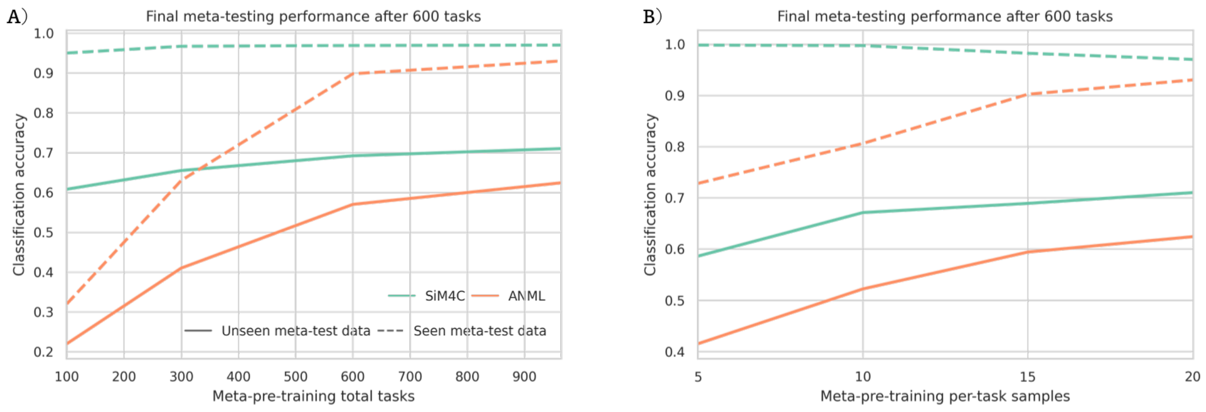

Figure 9. Performance on the online meta continual Omniglot benchmark *after meta pre-training* with reduced data-diversity. We compare the final meta-testing classification accuracy as we vary the **(A) the total number of meta-pre-training tasks** from 963 to $\{600, 300, 100\}$ and **(B) the samples available for each meta-pre-training task** from 20 to $\{15, 10, 5\}$.

due to the batch-normalization layers employed in the utilized ResNet architectures [19]. In contrast, we note that the fully-connected architectures in the Domain-IL experiments are unaffected by the variance of the inputs. We will refer to this approach as **SiM4C (split data)**. Lastly, we also consider a modification to this naive splitting strategy where, rather than random sampling, we partition sampled batch of data into D^{fut} and D^{cur} by computing the gradients with respect to the meta-learned parameters and find a partition that maximized the angle between their gradients. This strategy is inspired by recent methods that propose similar strategies for actively choosing which data to sample from the memory buffer [2]. The effects of splitting data with this criterion should be to induce a larger distribution shifts between D^{curr} and D^{fut}, trying to better approximate the distribution shift across different tasks. We will refer to this approach as **SiM4C (grad. split)**. To provide additional validation to the relevance of meta-learning in the auxiliary step, we also consider a simple ablation replacing SiM4C's exact second-order meta-optimization with an additional first-order gradient step on the same data with a different data augmentation. We refer to this baseline as **SiM4C (no meta)**.

We perform experiments comparing the efficacy of this ablation and the different strategies for obtaining future data in the S-CIFAR-10 benchmark for both Class-IL and Task-IL scenarios. We consider integrating these different versions of SiM4C both with the simple memory-based baseline from Riemer et al. [41] (ER), and Dark Experience Replay (DER) [6]. As reported in Table 8, all four versions of SiM4C appear to considerably improve performance from their base algorithm and the SiM4C (no meta) ablation. The only exceptions appears to be SiM4C with the split-based strategies when applied to DER on the Class-IL scenarios. In line with our previous intuition, this is likely caused by using batch normalization with inconsistent batch sizes between the past, current, and future data. This is evidenced by the fact that the same phenomenon does not occur when using SiM4C (split data/grad-split) with ER, since in the utilized base implementation (taken from [6]) past and current data are concatenated and evaluated together, as opposed to the base implementation of DER where they are processed separately. Furthermore, while in most of the considered settings, SiM4C (past data) provides the most benefits, its gains over the other strategies appear to be relatively marginal. Hence, our results appear to confirm the practicality and robustness of our approach, which we hope will contribute to shaping future meta-learning research toward empowering continual learning. As a final consideration, the gradient-based splitting strategy in SiM4C (grad. split) appears to provide overall improvements to SiM4C (split data), suggesting data selection strategies for meta-learning are a promising unexplored direction for future investigation.

References

[1] R. Aljundi, K. Kelchtermans, and T. Tuytelaars. Task-Free Continual Learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11246–11255, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.01151. 8

[2] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, 2019. 7, 8, 17, 21

[3] A. Antoniou, H. Edwards, and A. Storkey. How to train your maml. In *Seventh International Conference on Learning Representations*, 2019. 1, 3, 4, 19

[4] S. Beaulieu, L. Frati, T. Miconi, J. Lehman, K. O. Stanley, J. Clune, and N. Cheney. Learning to continually learn. In

ECAI 2020, pages 992–1001. IOS Press, 2020. 1, 2, 3, 4, 5, 8, 13, 14, 18

[5] A. S. Benjamin, D. Rolnick, and K. P. Kording. Measuring and regularizing networks in function space. *International Conference on Learning Representations*, 2019. 7, 8, 17

[6] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020. 2, 3, 7, 8, 15, 16, 17, 20, 21

[7] M. Caccia, P. Rodriguez, O. Ostapenko, F. Normandin, M. Lin, L. Page-Caccia, I. H. Laradji, I. Rish, A. Lacoste, D. Vázquez, and L. Charlin. Online Fast Adaptation and Knowledge Accumulation (OSAKA): A New Approach to Continual Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 16532–16545. Curran Associates, Inc., 2020. 1, 3

[8] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2019. 7, 8, 17

[9] A. Chaudhry, A. Gordo, P. K. Dokania, P. Torr, and D. Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. *arXiv preprint arXiv:2002.08165*, 2020. 7, 8, 17

[10] M. De Lange and T. Tuytelaars. Continual Prototype Evolution: Learning Online From Non-Stationary Data Streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8250–8259, 2021. 8

[11] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021. 1, 7, 15

[12] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni. Don't forget, there is more than forgetting: New metrics for Continual Learning. (Nips), 2018. 1, 8

[13] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 3, 4, 12, 14

[14] R. M. French. Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. In *Proceedings of the 13th annual cognitive science society conference*, volume 1, pages 173–178, 1991. 1, 8, 13

[15] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, Apr. 1999. ISSN 1364-6613. doi: 10.1016/S1364-6613(99) 01294-2. 1, 8

[16] S. T. Grossberg. *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, volume 70. Springer Science & Business Media, 2012. 1

[17] G. Gupta, K. Yadav, and L. Paull. Look-ahead meta learning for continual learning. *Advances in Neural Information Processing Systems*, 33:11588–11598, 2020. 1, 3, 4, 8, 18

[18] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020. 1

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7, 15, 21

[20] X. He, J. Sygnowski, A. Galashov, A. A. Rusu, Y. W. Teh, and R. Pascanu. Task Agnostic Continual Learning via Meta Learning. *arXiv:1906.05201 [cs, stat]*, June 2019. 8

[21] G. Hinton, O. Vinyals, and J. Dean. Dark knowledge. *Presented as the keynote in BayLearn*, 2(2), 2014. 3

[22] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 16

[23] K. Javed and M. White. Meta-learning representations for continual learning. *Advances in neural information processing systems*, 32, 2019. 1, 2, 3, 5, 8, 13, 14, 18

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 14

[25] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 7

[26] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7

[27] D. Kudithipudi, M. Aguilar-Simon, J. Babb, M. Bazhenov, D. Blackiston, J. Bongard, A. P. Brna, S. Chakravarthi Raja, N. Cheney, J. Clune, et al. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 4(3): 196–210, 2022. 1

[28] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 4, 5

[29] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 7

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 7

[31] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, June 2020. ISSN 1566-2535. doi: 10.1016/j.inffus.2019.12.004. 1, 8

[32] T. Lesort, A. Stoian, and D. Filliat. Regularization Shortcomings for Continual Learning. *arXiv:1912.03049 [cs, stat]*, Feb. 2020. 8

[33] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12), 2017. 16, 17

[34] V. Liu, R. Kumaraswamy, L. Le, and M. White. The utility of sparse representations for control in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4384–4391, 2019. 13

[35] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017. 7, 8, 12, 17

[36] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem.

In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1
[37] A. Nichol, J. Achiam, and J. Schulman. On First-Order Meta-Learning Algorithms, Oct. 2018. 8
[38] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 1, 12
[39] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019. 4
[40] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017. 7, 8, 17
[41] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019. 1, 2, 3, 7, 8, 12, 17, 18, 20, 21
[42] A. Rosasco, A. Carta, A. Cossu, V. Lomonaco, and D. Bacciu. Distilled Replay: Overcoming Forgetting Through Synthetic Samples. In F. Cuzzolin, K. Cannons, and V. Lomonaco, editors, *Continual Semi-Supervised Learning*, volume 13418, pages 104–117. Springer International Publishing, Cham, 2022. ISBN 978-3-031-17586-2 978-3-031-17587-9. doi: 10.1007/978-3-031-17587-9_8. 9
[43] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 16, 17
[44] M. Sangermano, A. Carta, A. Cossu, and D. Bacciu. Sample Condensation in Online Continual Learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08, July 2022. doi: 10.1109/IJCNN55064.2022.9892299. 9
[45] J. Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987. 3
[46] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, 2018. 16, 17
[47] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 14
[48] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning. *Arxiv preprint*, Apr. 2019. doi: 10.48550/arXiv.1904.07734. 1, 2
[49] G. M. van de Ven, H. T. Siegelmann, and A. S. Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11(1):4069, Aug. 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17866-2. 8
[50] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985. 2
[51] J. von Oswald, D. Zhao, S. Kobayashi, S. Schug, M. Caccia, N. Zucchet, and J. Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 5250–5263. Curran Associates, Inc., 2021. 8
[52] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros. Dataset Distillation. *arXiv:1811.10959 [cs, stat]*, Feb. 2020. 9
[53] F. Wiewel and B. Yang. Condensed Composite Memory Continual Learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2021. doi: 10.1109/IJCNN52387.2021.9533491. 9
[54] R. Yu, S. Liu, and X. Wang. Dataset Distillation: A Comprehensive Review, Jan. 2023. 9
[55] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, 2017. 7, 16, 17
[56] B. Zhao, K. R. Mopuri, and H. Bilen. Dataset Condensation with Gradient Matching. In *International Conference on Learning Representations*, Sept. 2020. 9